530-TP-001-001

Communications Subsystem DCE RPC/Socket Performance Analysis for the ECS Project

Technical Paper

Technical Paper—Not intended for formal review or government approval.

August 1995

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

H. Naveen /s/	8/11/95
Naveen Hota, Communications Engineer EOSDIS Core System Project	Date

SUBMITTED BY

Parag N. Ambardekar /s/	8/11/95
Parag Ambardekar, Release A Manager EOSDIS Core System Project	Date

Hughes Information Technology Corporation
Landover, Maryland

This page intentionally left blank.

Abstract

The Communications Subsystem (CSS) provides the overall communications infrastructure, and the communications services to support other subsystems in the Science and Communications Development Office (SCDO) and the Flight Operations Segment (FOS). This document contains a performance study on threaded DCE RPCs and Sockets, and a study on RPC performance with different Protection levels.

Keywords: CSMS, CSS, Communications, DCE, OODCE, Release A, RPC, Socket

This page intentionally left blank.

Contents

Abstract

1. Introduction

2. Threaded DCE RPCs vs. Socket Performance

3. RPC Performance with Different Protection Levels

Figures

Tables

Appendix A

1. Introduction

1.1 Purpose

This paper contains a performance study on threaded DCE RPCs and Sockets, and a study on RPC performance with different Protection levels.

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Corporation
1616 McCormick Dr.
Upper Marlboro, MD 20774

This page intentionally left blank.

2. Threaded DCE RPCs vs. Socket Performance

2.1 Objective

Describes the performance of threaded DCE RPCs and Sockets on the Local Area and Wide Area Network..

2.2 Methodology

- Various message sizes were transferred: 1K, 20K, 40K, 60K, 80K, and 100K bytes.
- HPs were used as platforms.
- Measurements were performed on two different media: LAN (Ethernet) and Internet (22 hops).
- In all cases multiple message transfers were conducted and averages were used for comparisons.
- The protocol used for threaded DCE RPCs was UDP/IP, and for sockets TCP/IP.
- The DCE tests were performed on machines on different DCE cells on the LAN and within the same DCE cell on the WAN.
- Response time is the elapsed time between just before the RPC or socket write call is made on the client side and just after the RPC or socket write is completed.

2.3 Overview of Threads

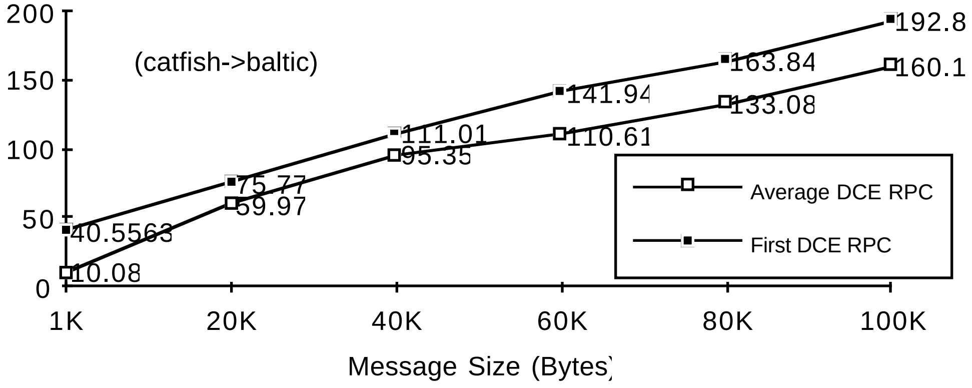
- DCE Threads is a user level thread package, it does not use the kernel resources.
- Threads are light weight processes that share a single address space and execute within it.
- Each thread shares all the resources of the originating process including signal handlers and descriptors, each thread has its own thread ID, thread specific data bindings and requires system resources to support a flow of control.
- Several threads are created within a process and execute concurrently. Within a multithreaded process there are at any time multiple points of execution.
- Threads improve throughput, computational speed and responsiveness of a program.
- Multiple threads improve performance on single processor system by permitting the overlap of input and output.
- The advantage of using multiple threads over using separate processes is that the former share a single address space, all open files and other resources.

2.4 Overview of Sockets

- Sockets are implemented within UNIX kernel, uses kernel resources.
- They consist of system calls that are entry point into the kernel.
- When fork is used in sockets, the system call causes creation of an exact clone of the caller's address space, resulting in the execution of two address spaces of the same code.

2.5 Statistical Analysis

Figure 1: Response Time Comparison of 1 Threaded DCE RPCs for Different Message Sizes on the LAN



Message Size	Average DCE RPC (ms)	First DCE RPC (ms)
1K	10.076	40.56
20K	59.97	75.77
40K	95.35	111.01
60K	110.61	141.94
80K	133.08	163.84
100K	160.11	192.80

First Threaded DCE RPC - Analysis of Variance:

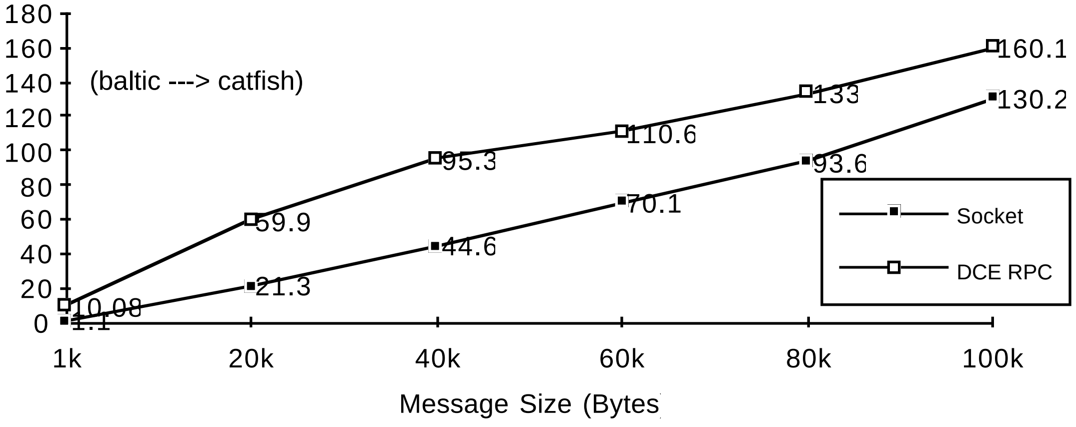
Source of Variation	*df*	*F*	*P-value*
Between Groups	5	230.85	0.001

Average Threaded DCE RPC - Analysis of Variance:

Source of Variation	*df*	*F*	*P-value*
Between Groups	5	1194.71	0.001

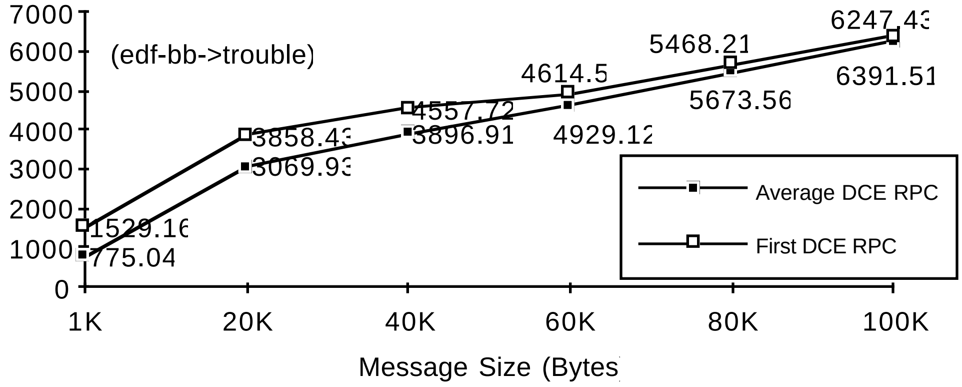
Figure 1 shows a statistical linear relationship exist between the dependent variable message size and the independent variable response time, that is, as the message size increases, the response time increases linearly. We separate the performance of the first DCE RPC between client and server from subsequent ones.

Figure 2: Response Time Comparison of Threaded DCE RPCs and Sockets for Different Message Sizes on the LAN



The results of Figure 2 show that sockets response time as a function of the message size increases linearly. The performance of sockets is faster than the threaded DCE RPC.

Figure 3: Response Time Comparison of 1 Threaded DCE RPCs for Different Message Sizes on the WAN



Message Size	Average DCE RPC (ms)	First DCE RPC (ms)
1K	775.04	1529.16
20K	3069.93	3858.43
40K	3896.91	4557.72
60K	4614.50	4929.12
80K	5468.21	5673.56
100K	6247.43	6391.51

First Threaded DCE RPC - Analysis of Variance

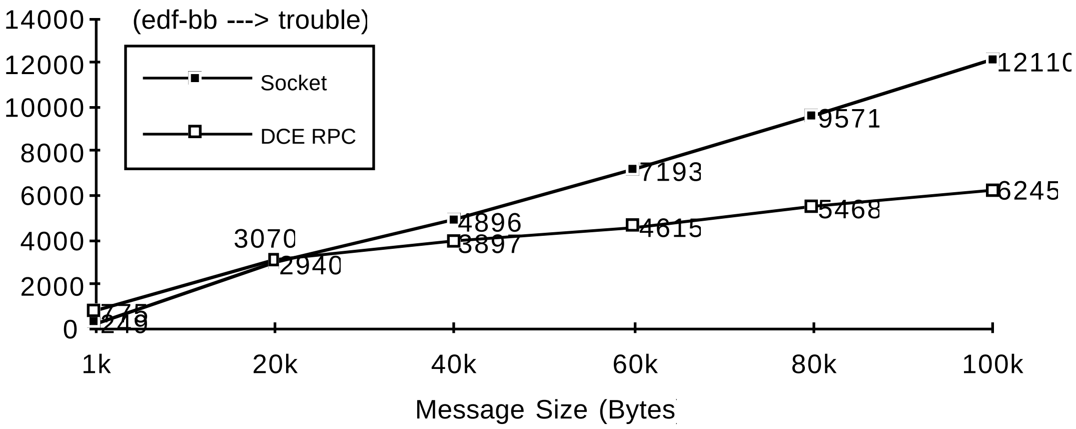
Source of Variation	*df*	*F*	*P-value*
Between Groups	5	347.01	0.001

Average Threaded DCE RPC - Analysis of Variance

Source of Variation	*df*	*F*	*P-value*
Between Groups	5	1256.74	P < 0.001

Figure 3 shows a statistical linear relationship exist between the dependent variable message size and the independent variable response time, that is, as the message size increases, the response time increases linearly. We separate the performance of the first threaded DCE RPC between client and server from subsequent ones. As the length of the message sent increases the difference between the first DCE RPC and the rest of them decreases, it becomes much less significant than on the LAN.

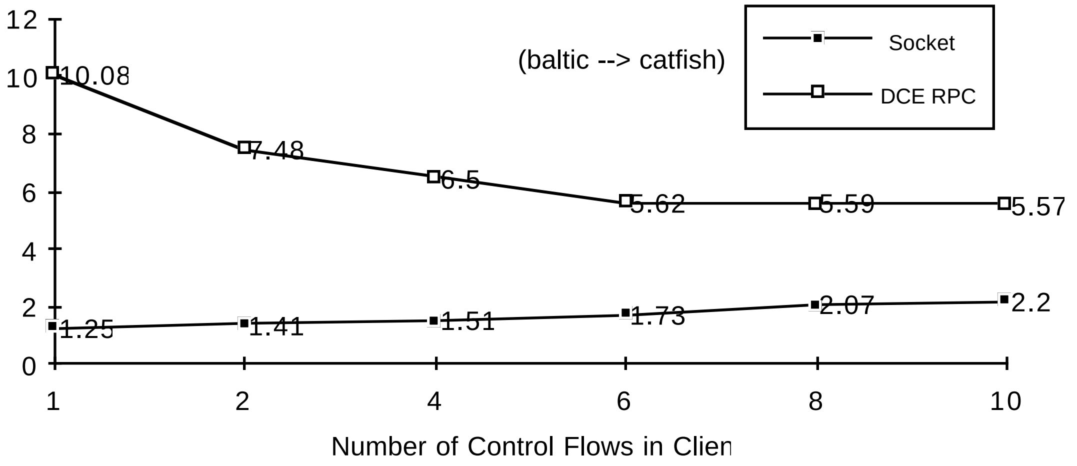
Figure 4: Response Time Comparison of Threaded DCE RPCs and Sockets for Different Message Size on the WAN



The results of Figure 4 show that sockets response time as a function of the message size increases more in comparison to DCE RPCs (when around 20 KB). This is due to the fact that sockets use TCP protocol, the overhead per packet is more overwhelming as more packets are sent since TCP provides end-to-end error detection and correction.

DCE RPCs use UDP, the connectionless datagram delivery service with no packet error detection, the error control is done at the message level.

Figure 5: Response Time Comparison of Threaded DCE RPCs and Sockets transferring 1K Byte on the LAN



Threaded DCE RPC Analysis of Variance

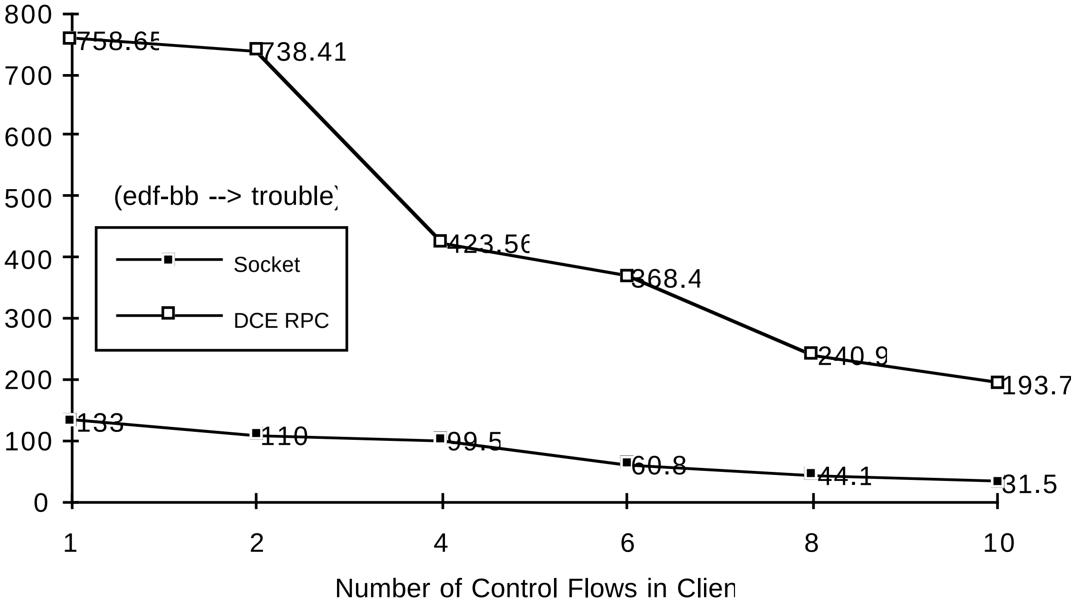
Source of Variation	*df*	*F*	*P-value*
Between Groups	3	280.68	P < 0.001

DCE RPC Correlation -0.7965

Figure 5 shows that statistically there is a linear inverse relationship between the dependent variable, the number of threads, and the independent variable, the response time, that is, as the number of threads increases, the response time decreases due to fewer context switches in the former that in the latter. The variation is more significant at first, when we go from 1 to 5 threads, than from 5 to 10 threads.

With respect to sockets, the response time increases as the number of processes increases, but these results only apply to Baltic and catfish.

Figure 6: Response Time Comparison of Threaded DCE RPCs and Sockets transferring 1K Byte on the WAN



Threads Analysis of Variance

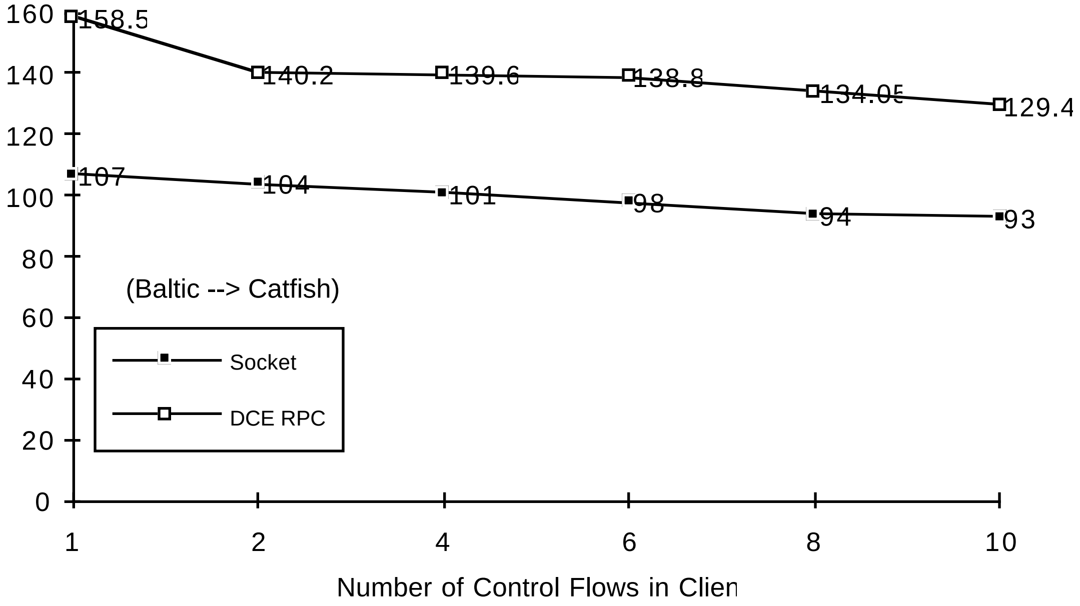
Source of Variation	*df*	*F*	*P-value*
Between Groups	3	3592.998	P < 0.001

Threads Correlation -0.9603

Figure 6 shows that statistically there is a linear inverse relationship between the dependent variable, the number of threads, and the independent variable, the response time, that is, as the number of threads increases, the response time decreases due to fewer context switches in the former that in the latter. The variation is more significant at first, when we go from 1 to 5 threads, than from 5 to 10 threads.

With respect to sockets, there is also a linear inverse relationship between the number of processes and the response time. Sockets perform better than DCE RPCs for all of the tests.

Figure 7: Response Time Comparison of Threaded DCE RPC and Sockets transferring 100K Bytes on the LAN



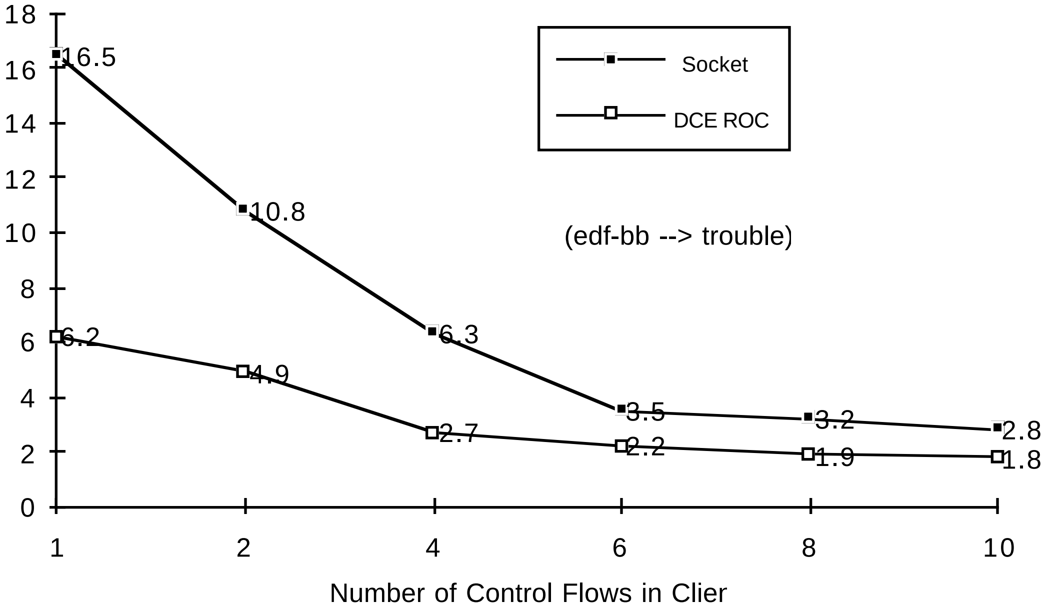
Threads Analysis of Variance

Source of Variation	df	F	P-value
Between Groups	3	25.534	P < 0.001

Threads Correlation -0.84

Figure 7 shows similar results as Figure 6. Please refer to the description of Figure 6.

Figure 8: Response Time Comparison of Threaded DCE RPCs and Sockets transferring 100K Bytes on the WAN



Threads Analysis of Variance

Source of Variation	*df*	*F*	*P-value*
Between Groups	3	1907.099	P < 0.001

Threads Correlation	-0.89

Figure 8 shows that statistically there is a linear inverse relationship between the dependent variable, the number of threads, and the independent variable, the response time, that is, as the number of threads increases, the response time decreases due to fewer context switches in the former that in the latter . The variation is more significant at first, when we go from 1 to 5 threads, than from 5 to 10 threads.

The average DCE RPC response time is lower than that of the socket due to the fact that sockets use TCP protocol, the overhead per packet is more overwhelming as more packets are sent since TCP provides end-to-end error detection and correction.

DCE RPCs use UDP, the connectionless datagram delivery service with no packet error detection, the error control is done at the message level.

2.6 Conclusion

From Figure 2, Figure 5, and Figure 7, we can conclude that on the LAN, sockets always perform better than DCE RPCs.

From Figure 4, and Figure 8, we can conclude that on the WAN, DCE RPCs perform better than sockets for large message size, that is over 20 KB.

3. RPC Performance with Different Protection Levels

3.1 Application Details

The application server was multi-threaded (to a default maximum of ten). The client composed the string binding handle using host address & protocol sequence. Once the binding is obtained the client makes an RPC transferring 1000 packets of 1600 bytes of data (ASCII character strings) each to the server in 10 threads. The server didn't do any computationally intensive tasks, it just receives the data and discards it.

The client was run 25 times, and the elapsed time to transfer the data was noted down and plotted as shown in the figures in Appendix A. The elapsed time includes the time to find end point (port) from the *rpcd* daemon for the communication to take place. As indicated in the figures, the following were studied:

- Socket
- Non-Authenticated RPCs
- rpc_protect_level_connect: Performs protection only when the client establishes a relationship with the server.
- rpc_protect_level_pkt: Ensures that all the data received is from the expected client.
- rpc_c_protect_level_pkt_integ: Ensures and verifies that none of the data transferred between client and server has been modified.
- rpc_c_protect_level_pkt_privacy: Performs protection as specified by all of the previous levels and also encrypt each RPC argument value.

3.2 csmscell.hitc.com

3.2.1 Test Environment

The performance analysis was done on HP-UX 9.05 9000/735 workstations running DCE 1.0.3. The experiments were carried out in the cell "csmscell.hitc.com". This cell consists of four HP workstations -- baltic, cyclops, london, and york (where baltic served as the DCE server). All the workstations were in an Ethernet LAN.

The application client and server were run on same and different machines, these machines were different from the machine that served as the DCE server. The experiments were carried out with UDP/IP as the network protocol. Since the client and server machines were the same type, there was little data marshaling/unmarshaling overhead in our results.

3.2.2 Discussion

The performance of RPCs is poor within the same machine (single processor) due to the context switching[1] time (of the CPU) between the client and server processes.

On the average, RPCs with no security features between different machines, is four times slower than sockets (table 3.2.2.-2).

RPCs with the highest protection level, is four times slower than RPCs with no security features (table 3.2.2.-3).

Table 3.2.2-1. Performance of Socket & RPCs (with different protection levels) (1 of 2)

	Socket	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
london.hitc.com to london.hitc.com						
Average time in seconds	0.7	6.6	10.4	11.1	13.5	23
Socket::	1	9.4	14.9	15.9	19.3	32.9
Non-Authenticated RPCs::		1	1.6	1.6	2.1.1	3.5
york.hitc.com to york.hitc.com						
Average time in seconds	0.2	2.6	4.1	4.1	6.4	10.4
Socket::	1	13	20.5	20.5	32	52
Non-Authenticated RPCs::		1	1.6	1.6	2.4	4
cyclops.hitc.com to cyclops.hitc.com						
Average time in seconds	0.3	2.8	4.4	4.4	6.7	11.3
Socket::	1	9.3	14.6	14.6	22.3	37.6
Non-Authenticated RPCs::		1	1.6	1.6	2.4	4
london.hitc.com to york.hitc.com						
Average time in seconds	0.9	3.5	4.9	4.9	6.9	10.3
Socket::	1	3.9	5.4	5.4	7.7	11.4
Non-Authenticated RPCs::		1	1.4	1.4	2	3.0

[1] In a uniprocessor machine, the computer runs one process for a short period of time and then switches to anothee. Changing from one process to another is called context switch.

Table 3.2.2-1. Performance of Socket & RPCs (with different protection levels) (2 of 2)

	Socket	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
london.hitc.com to cyclops.hitc.com						
Average time in seconds	0.9	3.5	5.3	5.6	7.6	11
Socket::	1	3.9	5.9	6.2	8.4	12.2
Non-Authenticated RPCs::		1	1.5	1.6	2.2	3.2
york.hitc.com to cyclops.hitc.com						
Average time in seconds	0.6	2.3	3.3	3.3	5.5	9.2
Socket::	1	3.8	5.5	5.5	9.2	15.3
Non-Authenticated RPCs::		1	1.5	1.5	2.3	4

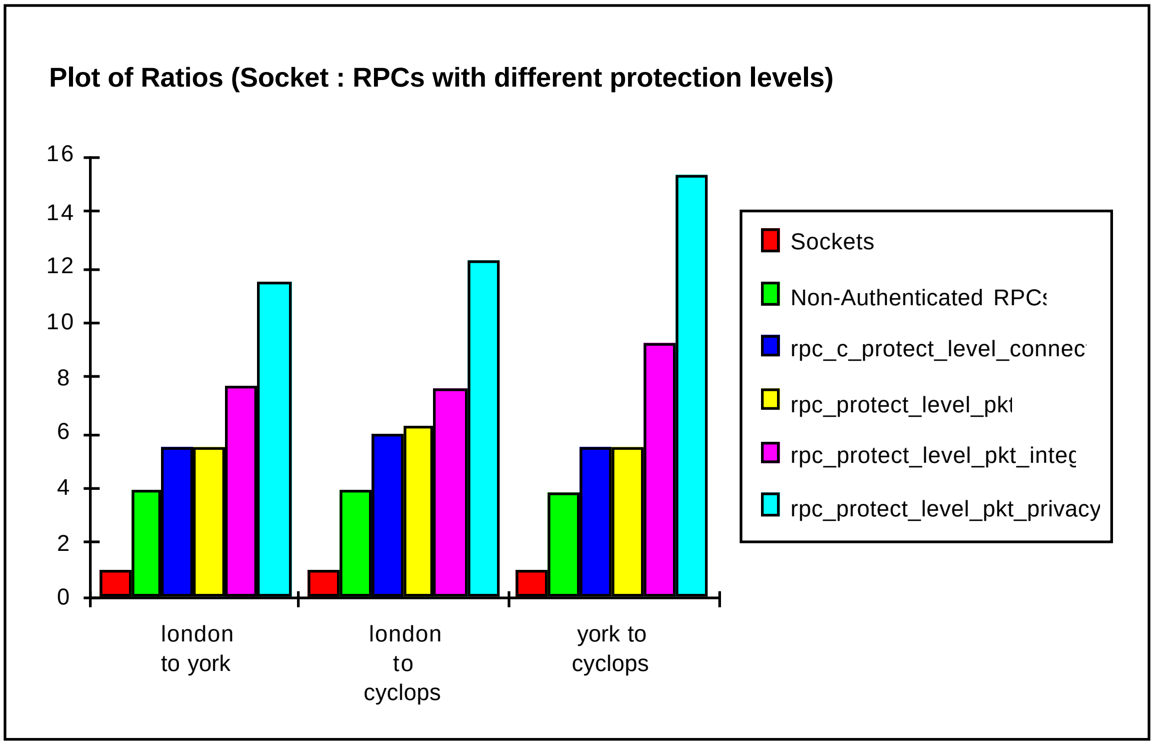


Figure 3.2.2-1. Socket : RPCs with different protection levels

Table 3.2.2-2. Average of Ratios (Socket : RPCs with different protection levels) (client & server on different machines)

	Socket	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
Average Ratio	1	3.9	5.6	5.7	8.4	13

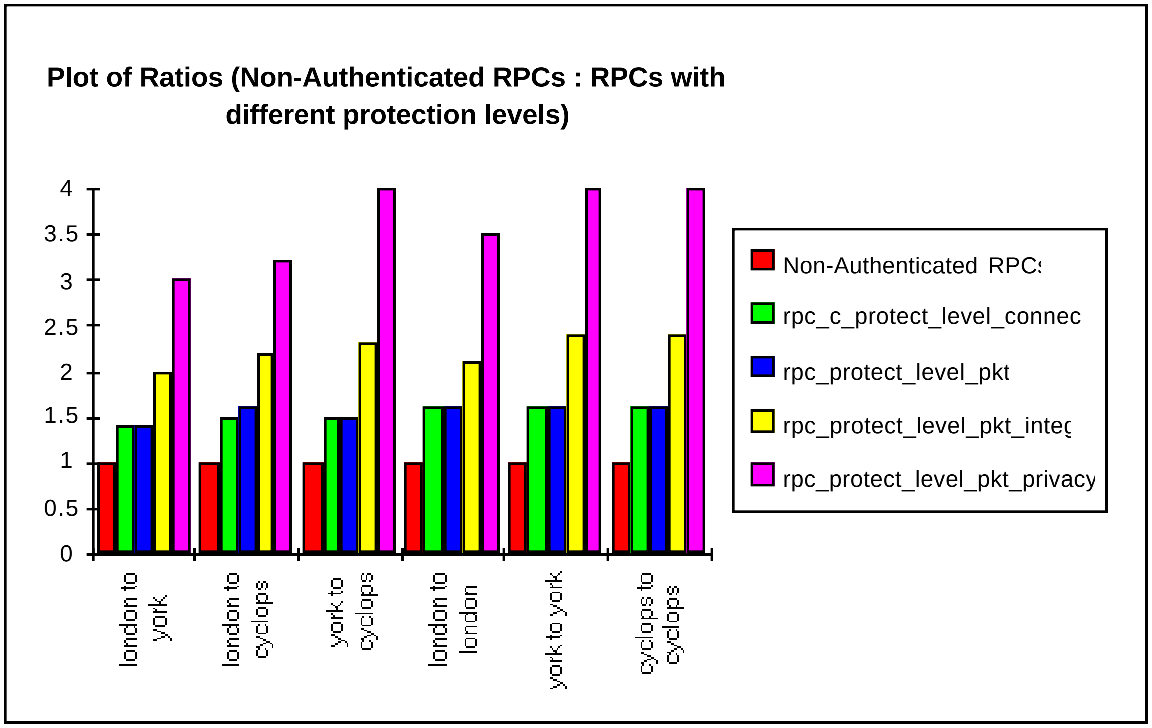


Figure 3.2.2-2 Non-Authenticated RPCs: RPCs with different protection levels

Table 3.2.2-3. Average of the Ratios (Non-Authenticated RPCs: RPCs with different protection levels)

	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
Average Ratio	1	1.5	1.6	2.2	3.6

3.3 edfcell.hitc.com

3.3.1 Test Environment

The performance analysis was done on the workstations listed in Table 3.3.1-1.

Table 3.3.1-1. Machines Used in edf-cell.hitc.com

Hostname	Vendor / Version	DCE Version
boston.hitc.com	DEC OSF1 v3.0	OSF DCE 1.0.3
catfish.hitc.com	HP-UX 9.01 9000/715	OSF DCE 1.0.2
csms2.hitc.com	SunOS 2.3	OSF DCE 1.0.3
fire.hitc.com	SunOS 2.3	OSF DCE 1.0.3

3.3.2 Discussion

Table 3.3.2-1. Performance of Socket & RPCs (with different protection levels) (1 of 2)

	Socket	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
caspian.hitc.com to catfish.hitc.com						
Average time in seconds	1.7	7.4	9.6	9.6	11.4	–
Socket::	1	4.4	5.6	5.6	6.7	–
Non-Authenticated RPCs::		1	1.3	1.3	1.5	–
csms2.hitc.com to fire.hitc.com						
Average time in seconds	0.6	10.2	14.6	15	18.5	–
Socket::	1	17	24	25	31	–
Non-Authenticated RPCs::		1	1.4	1.5	1.8	–
catfish.hitc.com to csms2.hitc.com						
Average time in seconds	6.5	8	12.6	13.3	16.9	–
Socket::	1	1.3	1.9	2	2.6	–
Non-Authenticated RPCs::		1	1.6	1.7	2.1	–
fire.hitc.com to catfish.hitc.com						
Average time in seconds	1.8	12.3	15.5	16.6	22	–
Socket::	1	6.8	8.6	9.2	12.2	–
Non-Authenticated RPCs::		1	1.3	1.4	1.8	–
boston.hitc.com to fire.hitc.com						
Average time in seconds	0.5	9.1	13.5	13.6	16.5	–
Socket::	1	18.2	27	27.2	33	–
Non-Authenticated RPCs::		1	1.5	1.5	1.8	–

Table 3.3.2-1. Performance of Socket & RPCs (with different protection levels) (2 of 2)

	Socket	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
boston.hitc.com to catfish.hitc.com						
Average time in seconds	1	4.9	7.3	7.4	10.6	–
Socket::	1	4.9	7.3	7.4	10.6	–
Non-Authenticated RPCs::		1	1.5	1.5	2.2	–

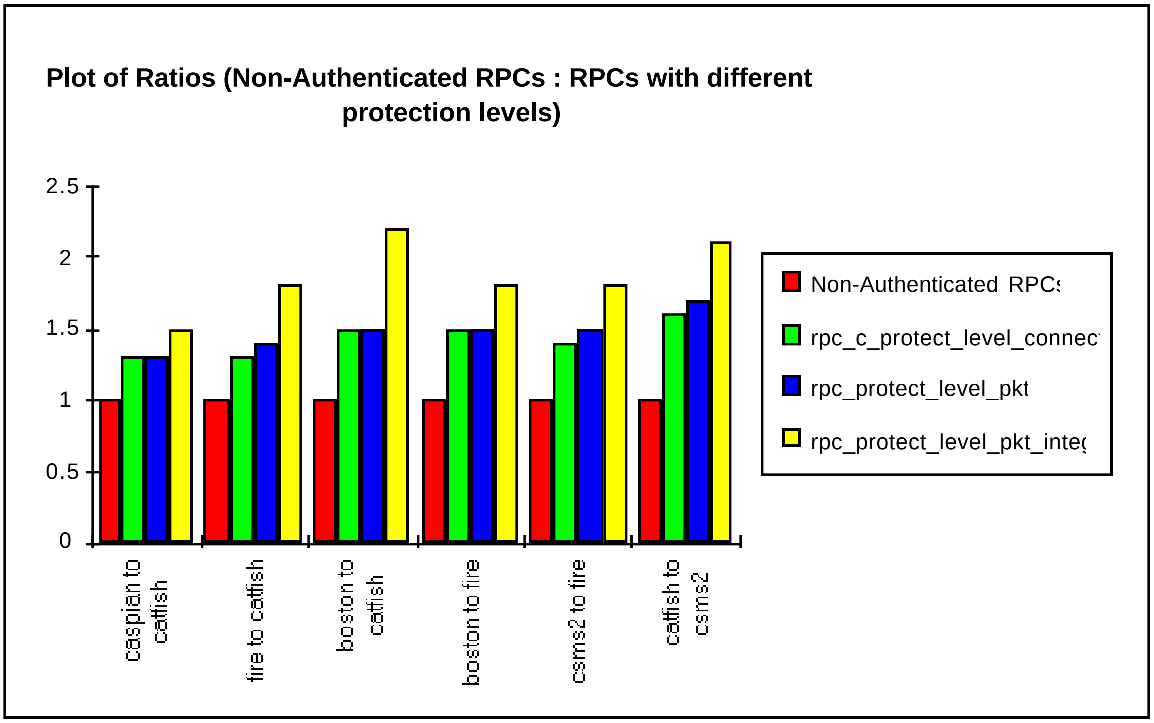


Figure 3.3.2-1. Non-Authenticated RPCs: RPCs with different protection levels

Table 3.3.2-2. Average of the Ratios (Non-Authenticated RPCs : RPCs with different protection levels)

	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
Average Ratio	1	1.4	1.5	1.9	–

3.4 epcell.hitc.com

3.4.1 Test Environment

3.4.2 Discussion

Table 3.4.2-1. Performance of Socket & RPCs (with different protection levels) (1 of 2)

	Socket	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
edf-bb.gsfc.nasa.gov to edf-bb.gsfc.nasa.gov						
Average time in seconds	1	10.3	11.8	12	19.6	25.2
Socket::	1	10.3	11.8	12	19.6	25.2
Non-Authenticated RPCs::		1	1.2	1.2	1.9	2.5
edf-bb.gsfc.nasa.gov to epserver.gsfc.nasa.gov						
Average time in seconds	1.5	5.1	5.9	6	9.6	13.1
Socket::	1	3.4	3.9	4	6.4	8.7
Non-Authenticated RPCs::		1	1.2	1.2	1.9	2.6
edf-bb.gsfc.nasa.gov to ecs-hp1.cr.usgs.gov						
Average time in seconds	61.1	103.8	99.8	105.4	106	107.9
Socket::	1	1.7	1.6	1.7	1.7	1.8
Non-Authenticated RPCs::		1	~1	~1	~1	~1
ecsgsfc1.gsfc.nasa.gov to epserver.gsfc.nasa.gov						
Average time in seconds	13.8	19.9	19.9	20.1	20.9	20.7
Socket::	1	1.4	1.4	1.5	1.5	1.5
Non-Authenticated RPCs::		1	~1	~1	~1	~1
snowfall.colorado.edu to epserver.gsfc.nasa.gov						
Average time in seconds	52.9	68.7	72.3	72.3	83.2	86.1
Socket::	1	1.3	1.4	1.4	1.6	1.6
Non-Authenticated RPCs::		1	1.1	1.1	1.2	1.2

Table 3.4.2-1. Performance of Socket & RPCs (with different protection levels) (2 of 2)

	Socket	Non-Authenticated RPCs	rpc_c_protect_level_connect	rpc_c_protect_level_pkt	rpc_c_protect_level_pkt_integ	rpc_c_protect_level_pkt_privacy
trouble.gi.alaska.edu to epserver.gsfc.nasa.gov						
Average time in seconds	95	136.8	143.3	142.9	142.1	143.5
Socket::	1	1.4	1.5	1.5	1.5	1.5
Non-Authenticated RPCs::		1	~1	~1	~1	~1
trouble.gi.alaska.edu to trouble.gi.alaska.edu						
Average time in seconds	0.8	9.6	11	11	18.5	24.3
Socket::	1	12	13.8	13.8	23.1	30.4
Non-Authenticated RPCs::		1	1.2	1.2	1.9	2.5

Appendix A

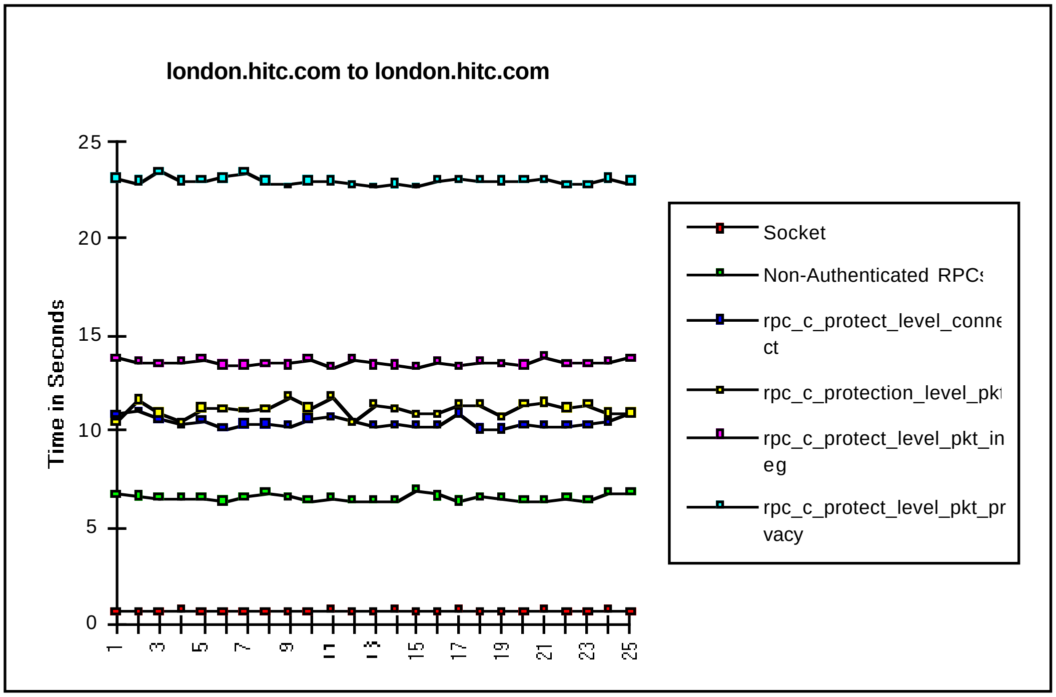


Figure A-1. london.hitc.com to london.hitc.com

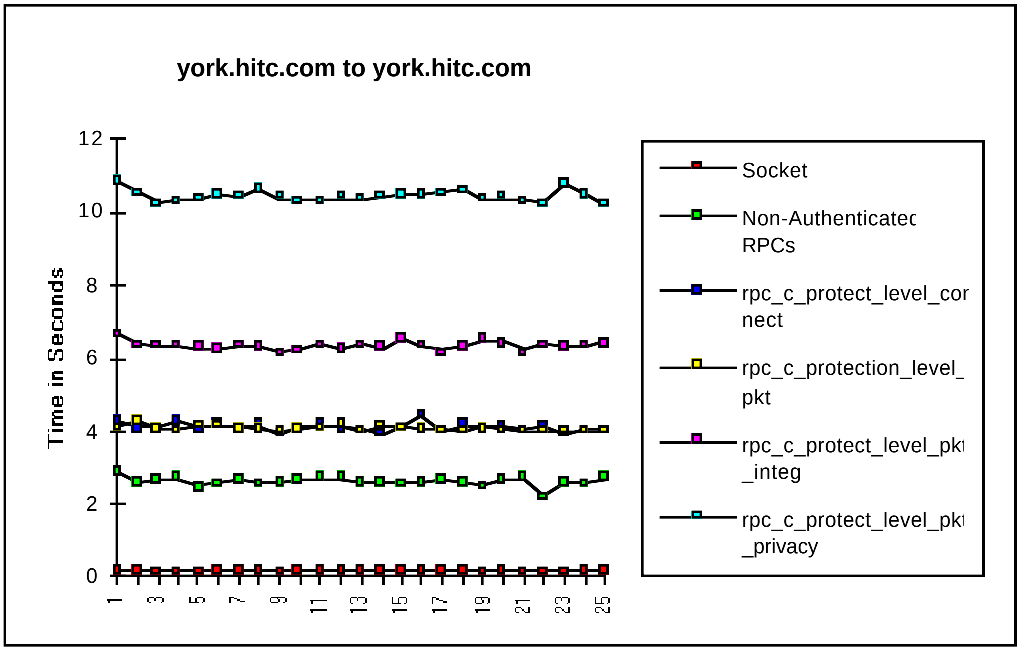


Figure A-2. york.hitc.com to york.hitc.com

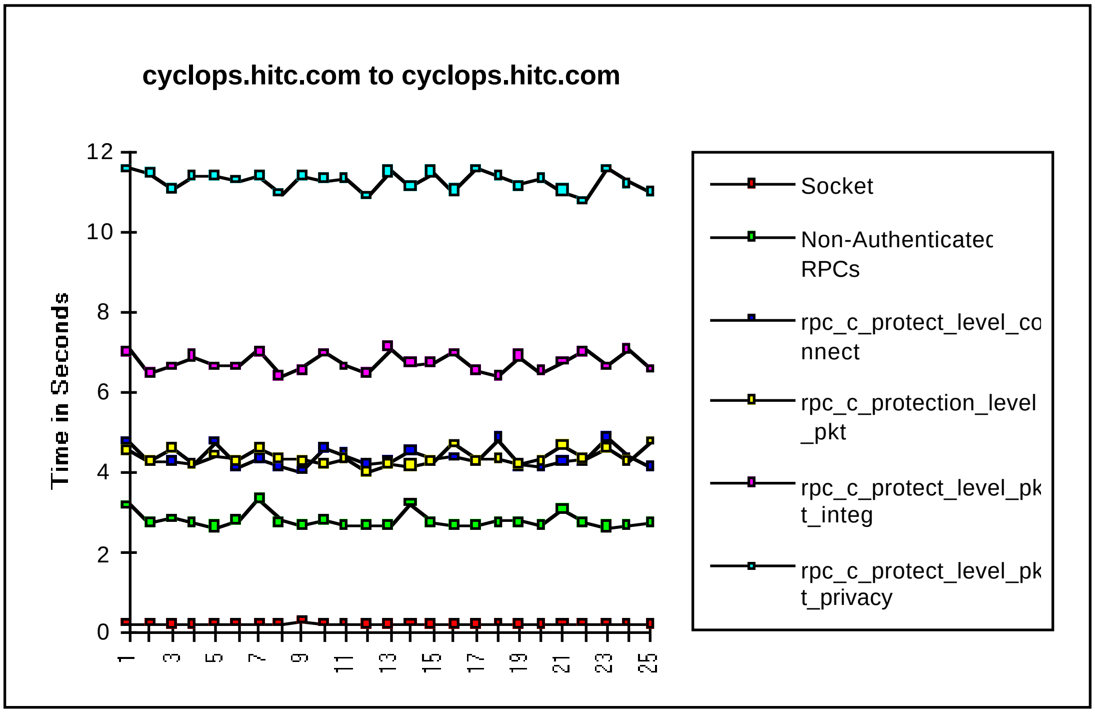


Figure A-3. cyclops.hitc.com to cyclops.hitc.com

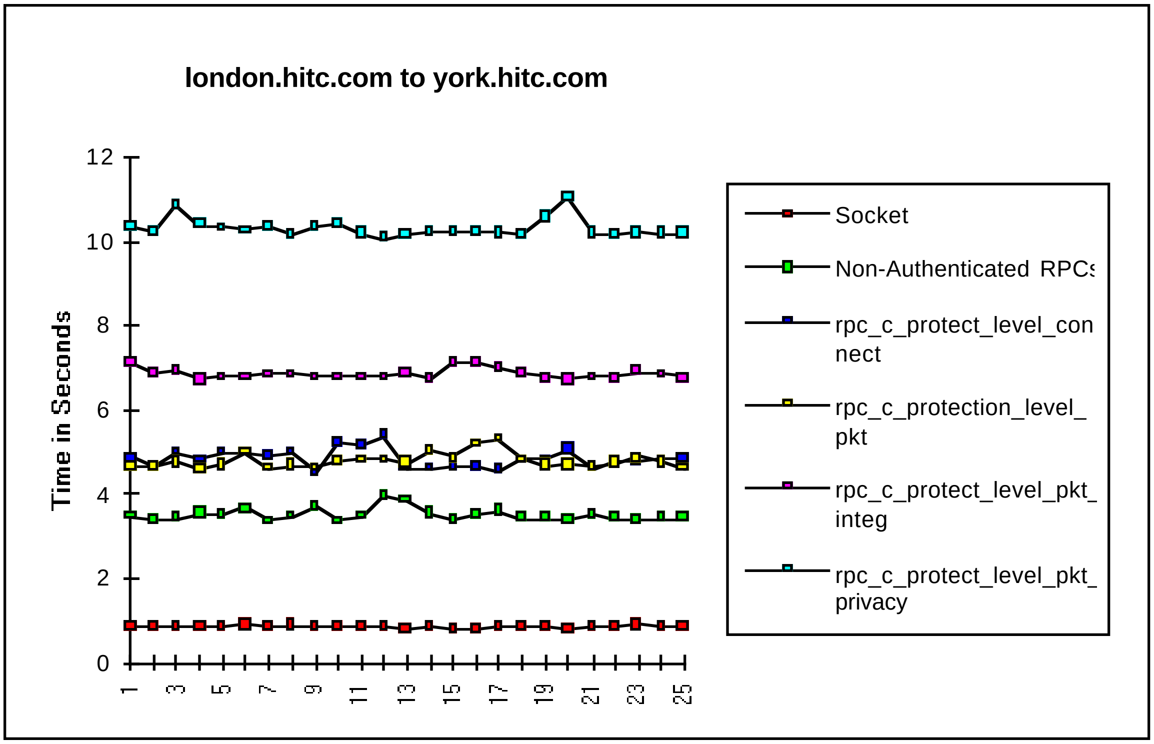


Figure A-4. london.hitc.com to york.hitc.com

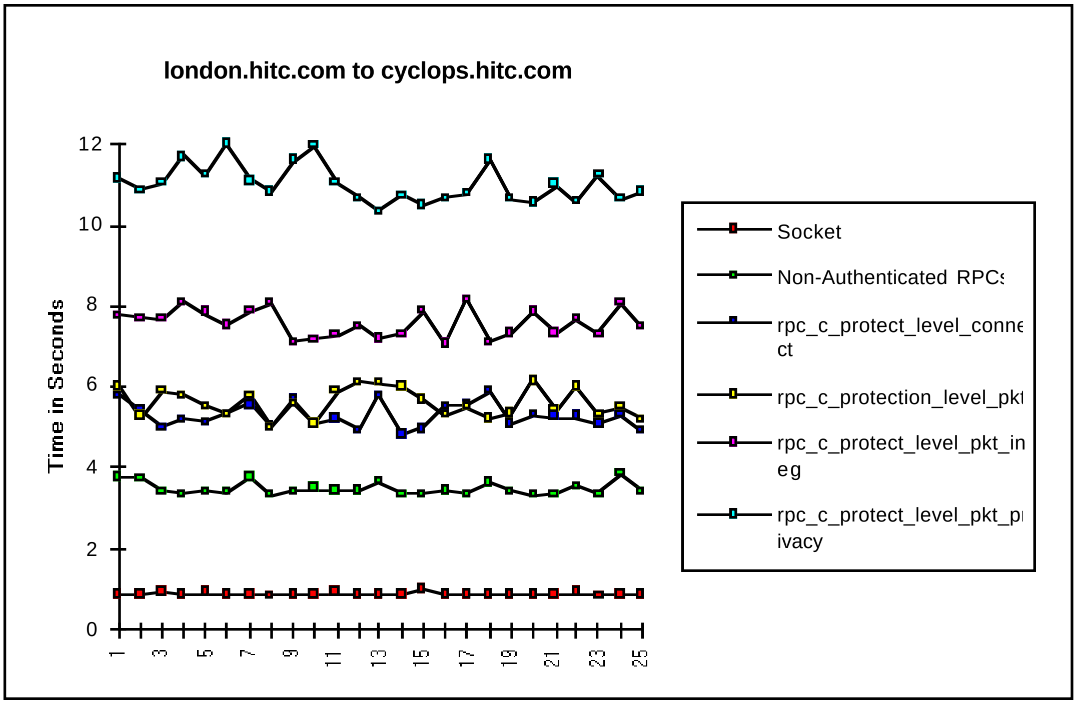


Figure A-5. london.hitc.com to cyclops.hitc.com

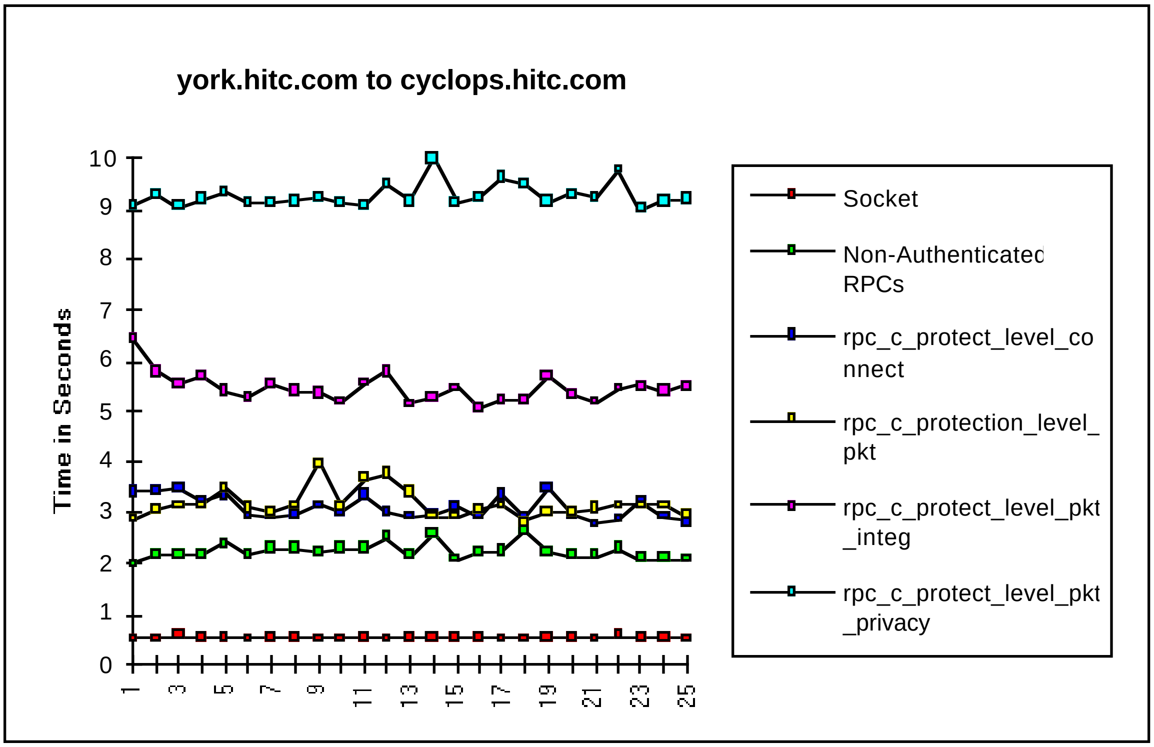


Figure A-6. york.hitc.com to cyclops.hitc.com

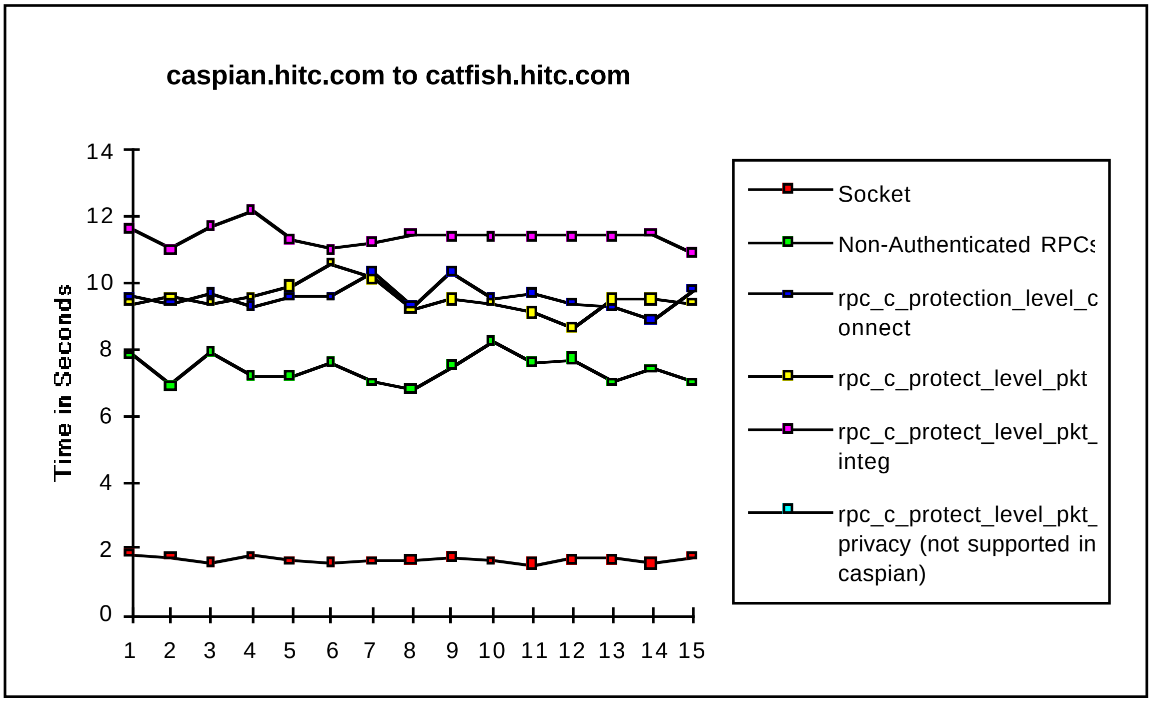


Figure A-7. caspian.hitc.com to catfish.hitc.com

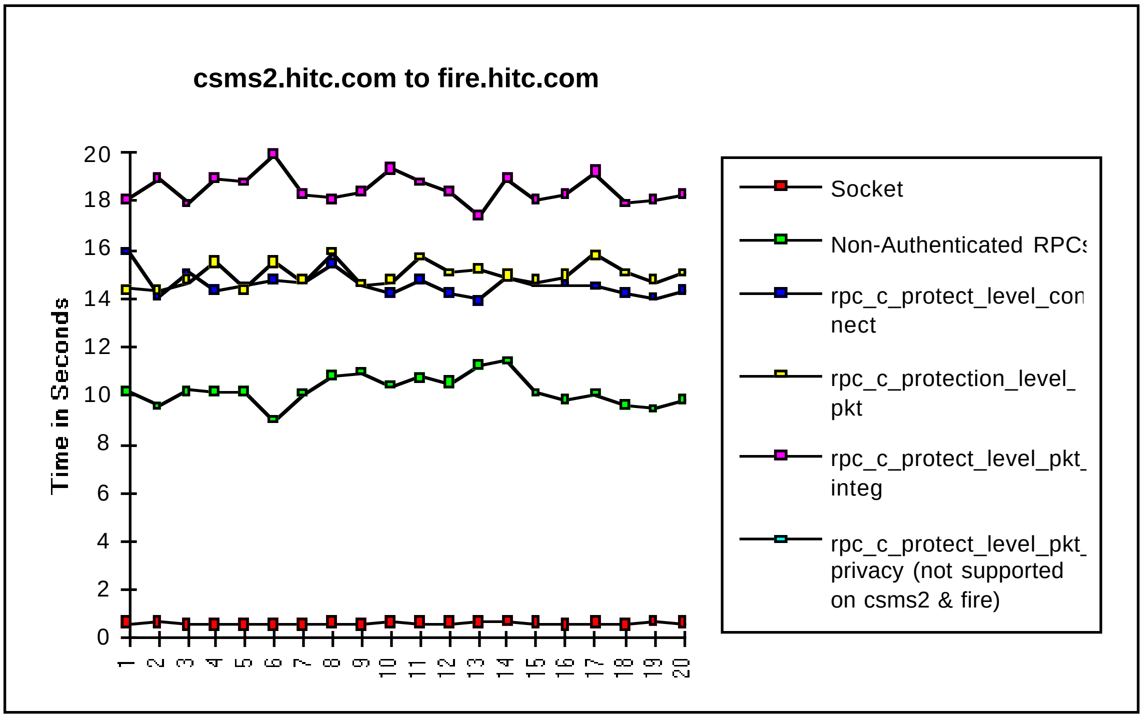


Figure A-8. csms2.hitc.com to fire.hitc.com

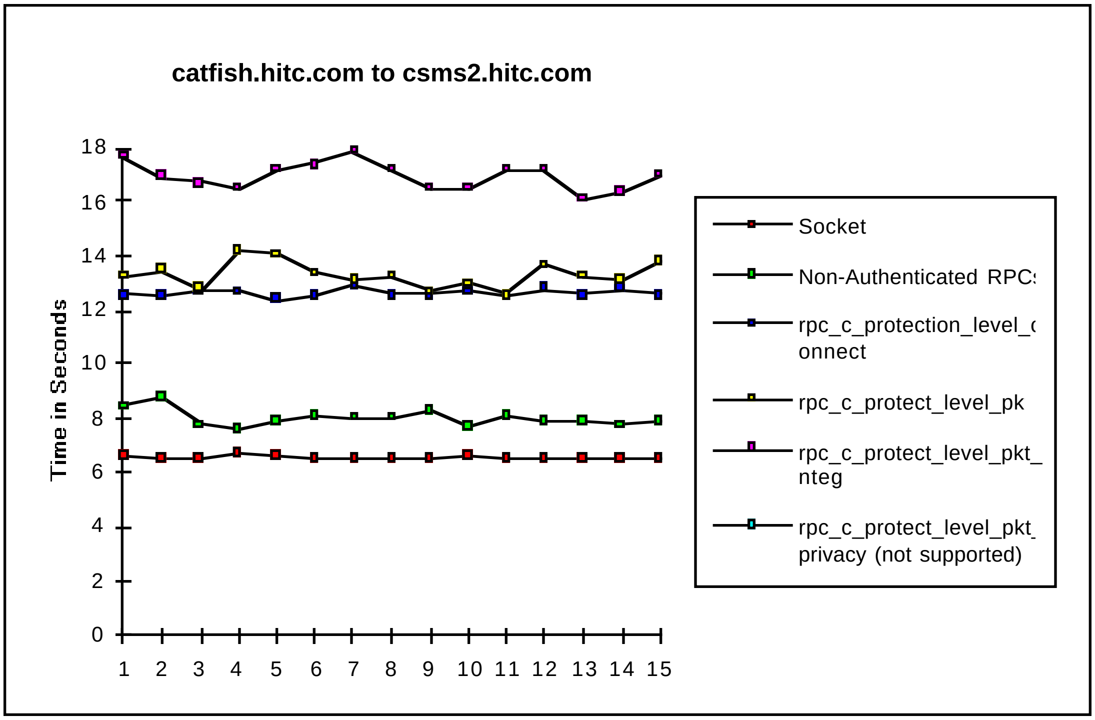


Figure A-9. catfish.hitc.com to csms2.hitc.com

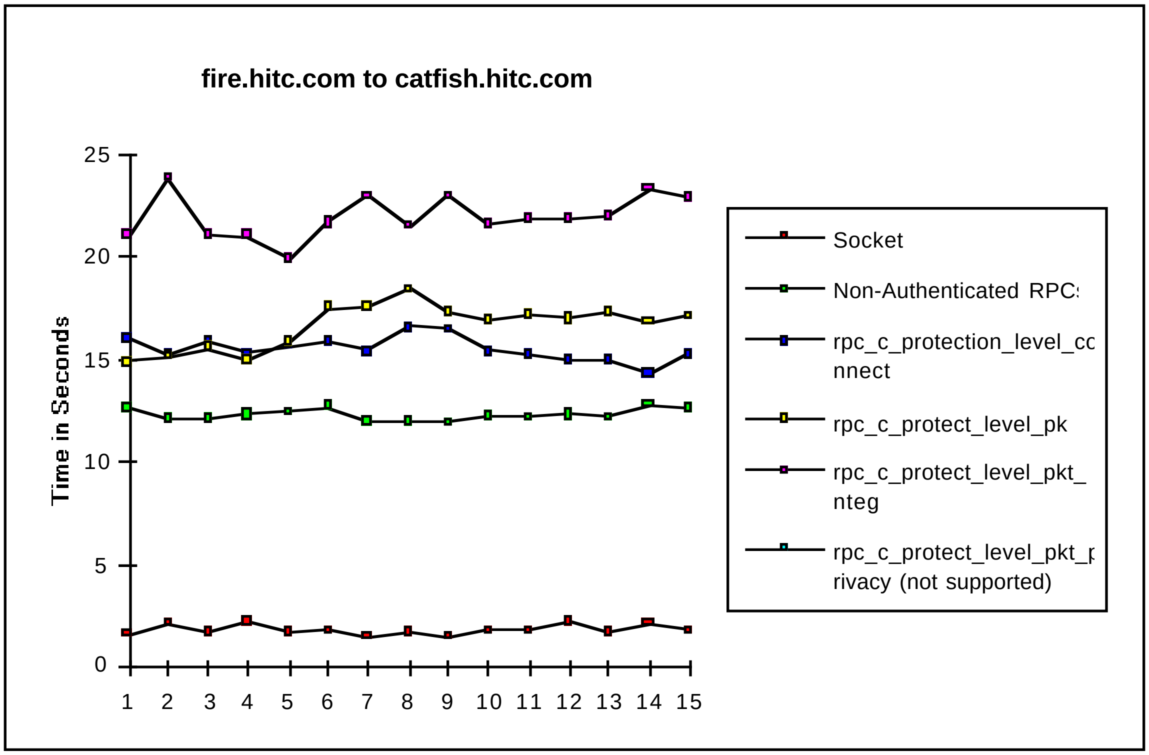


Figure A-10. fire.hitc.com to catfish.hitc.com

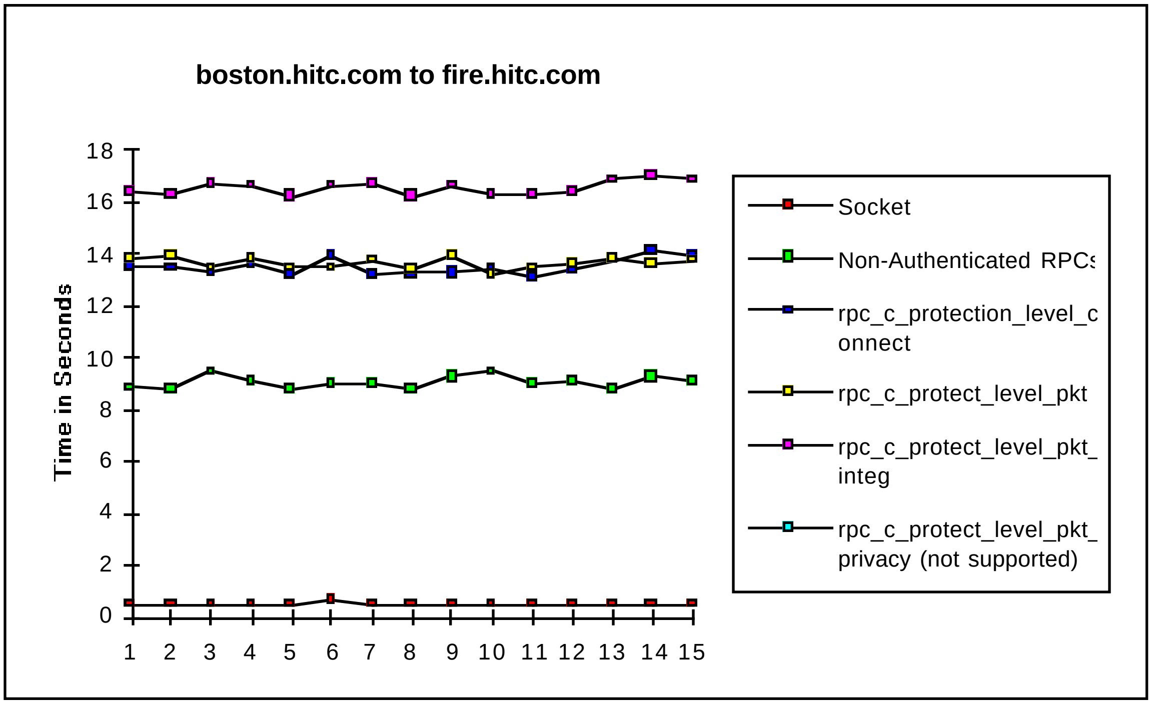


Figure A-11. boston.hitc.com to fire.hitc.com

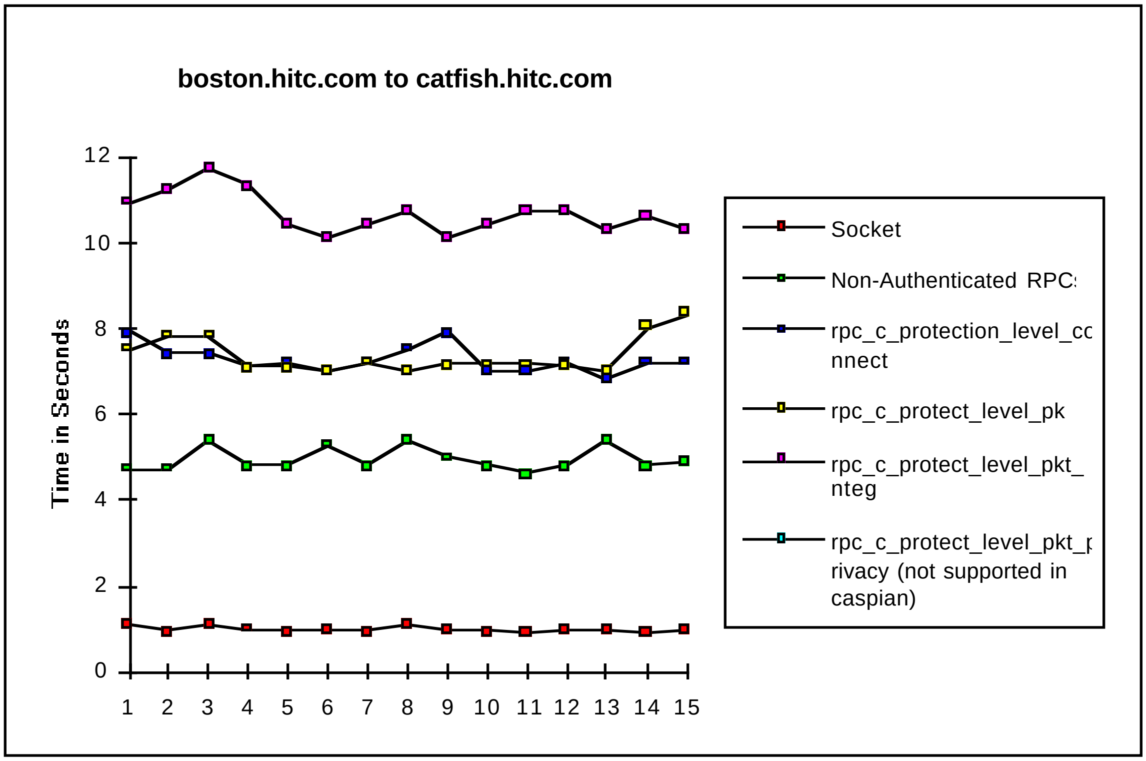


Figure A-12. boston.hitc.com to catfish.hitc.com

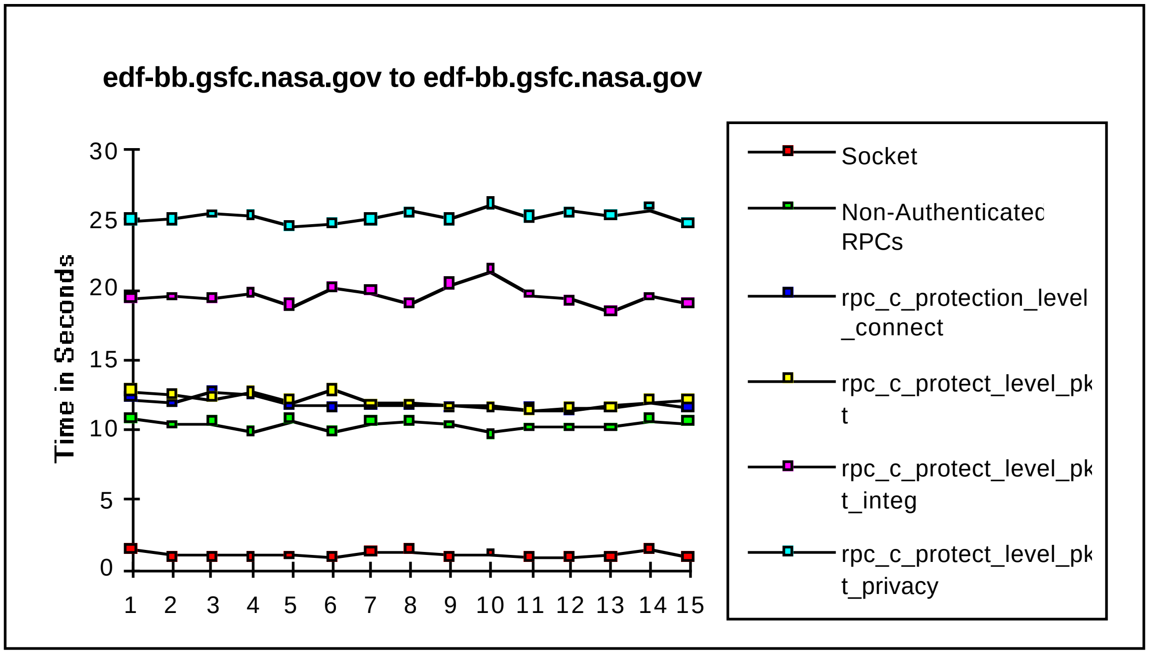


Figure A-13. edf-bb.gsfc.nasa.gov to edf-bb.gsfc.nasa.gov

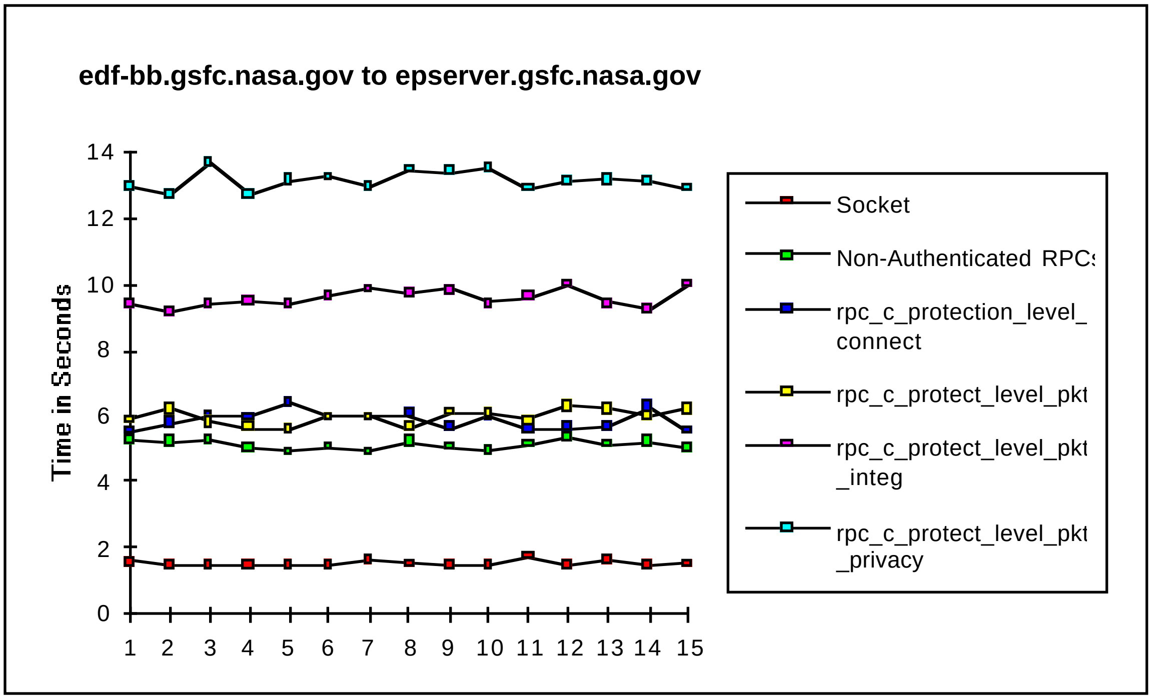


Figure A-14. edf-bb.gsfc.nasa.gov to epserver.gsfc.nasa.gov

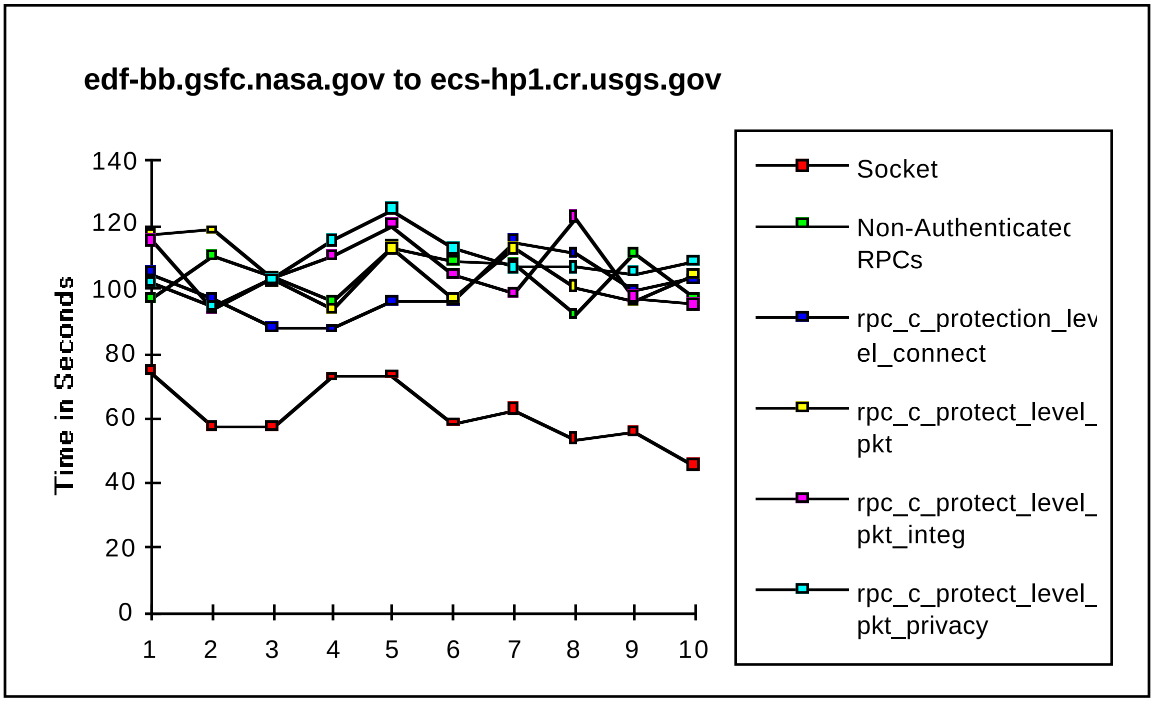


Figure A-15. edf-bb.gsfc.nasa.gov to ecs-hp1.cr.usgs.gov

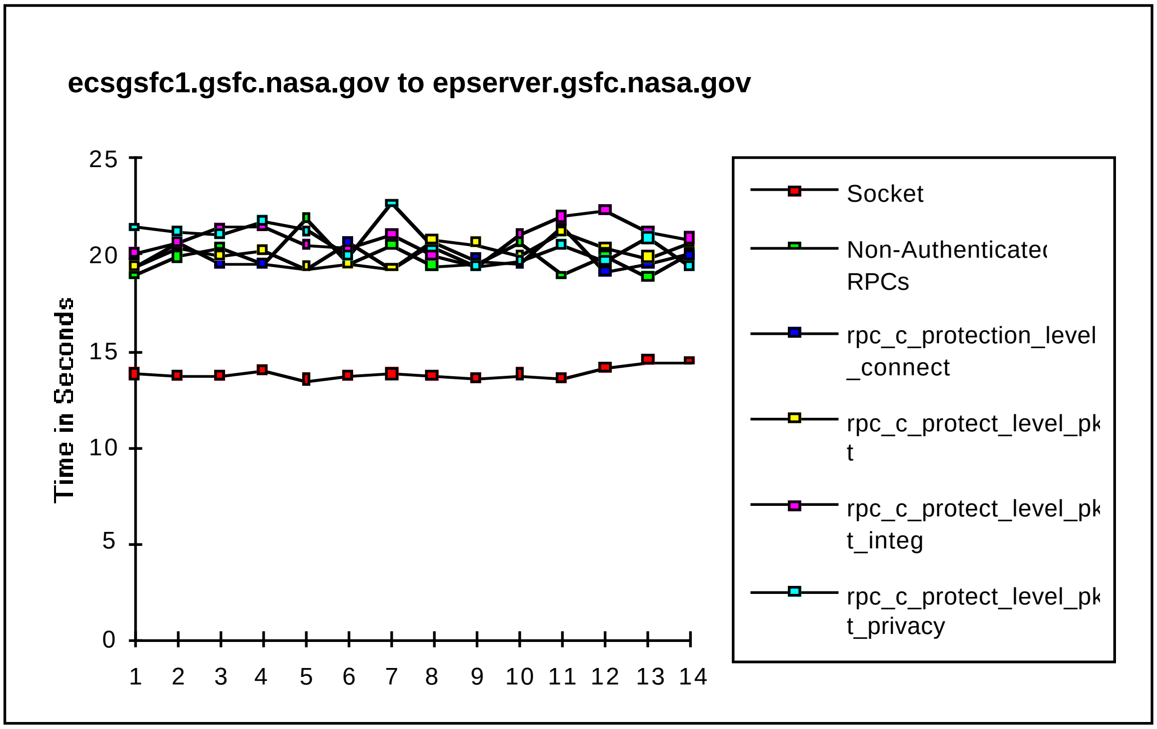


Figure A-16. ecsgsfc1.gsfc.nasa.gov to epserver.gsfc.nasa.gov

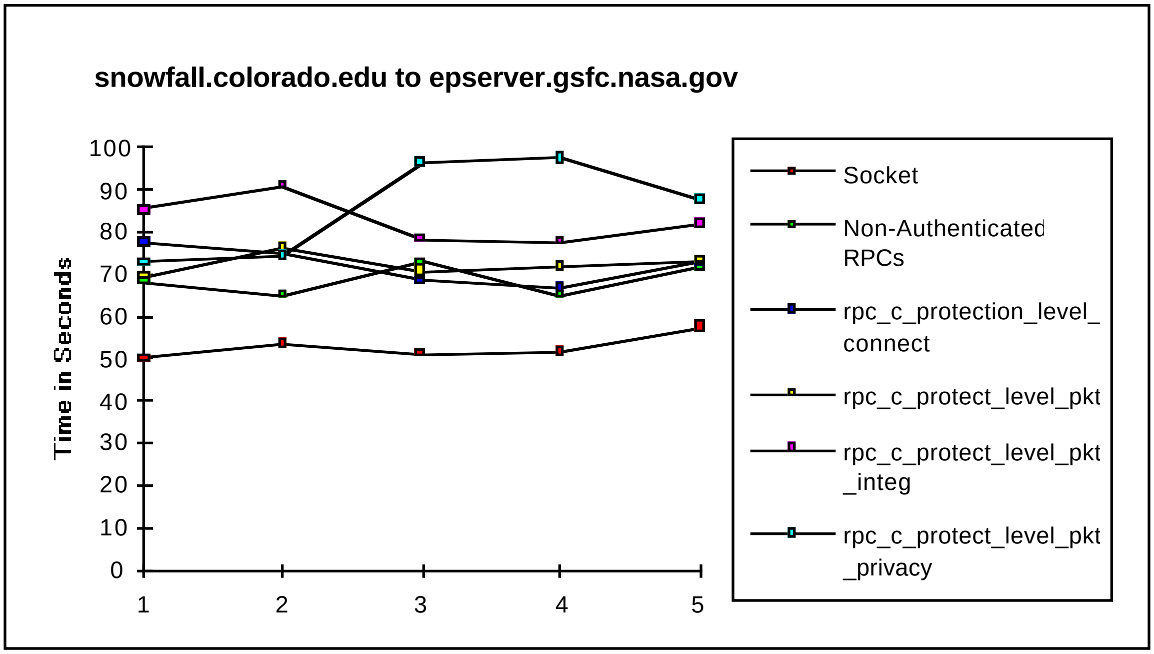


Figure A-17. snowfall.colorado.edu to epserver.gsfc.nasa.gov

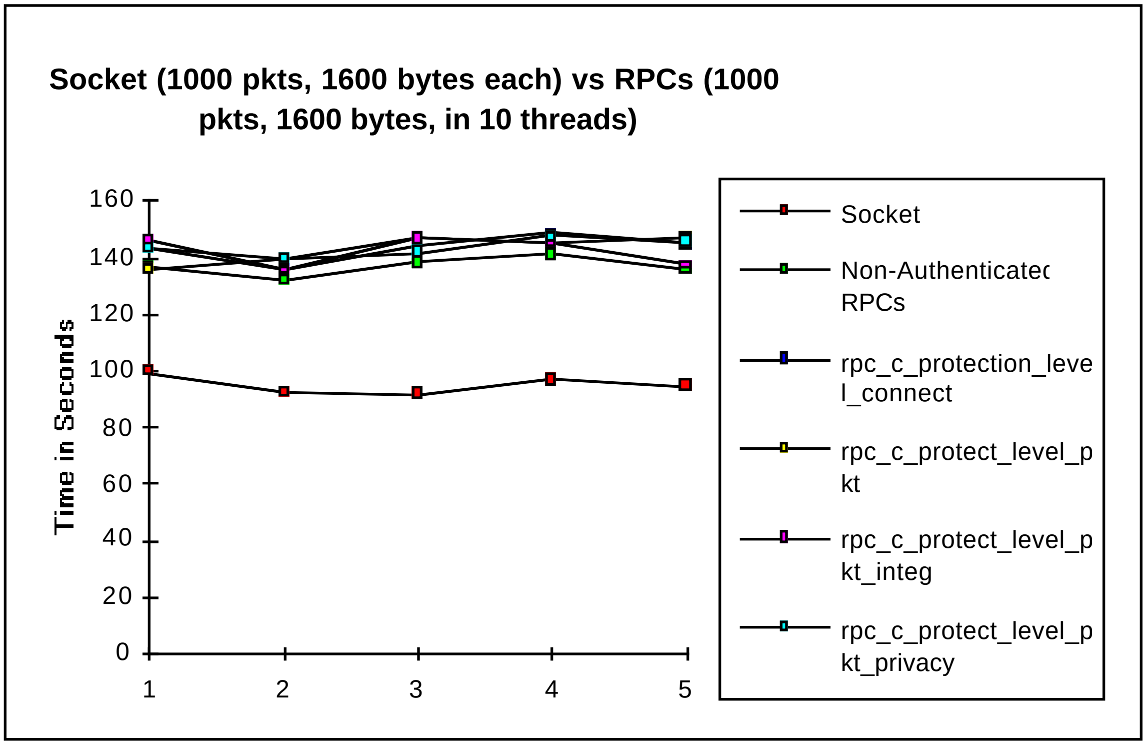


Figure A-18. trouble.gi.alaska.edu to epserver.gsfc.nasa.gov

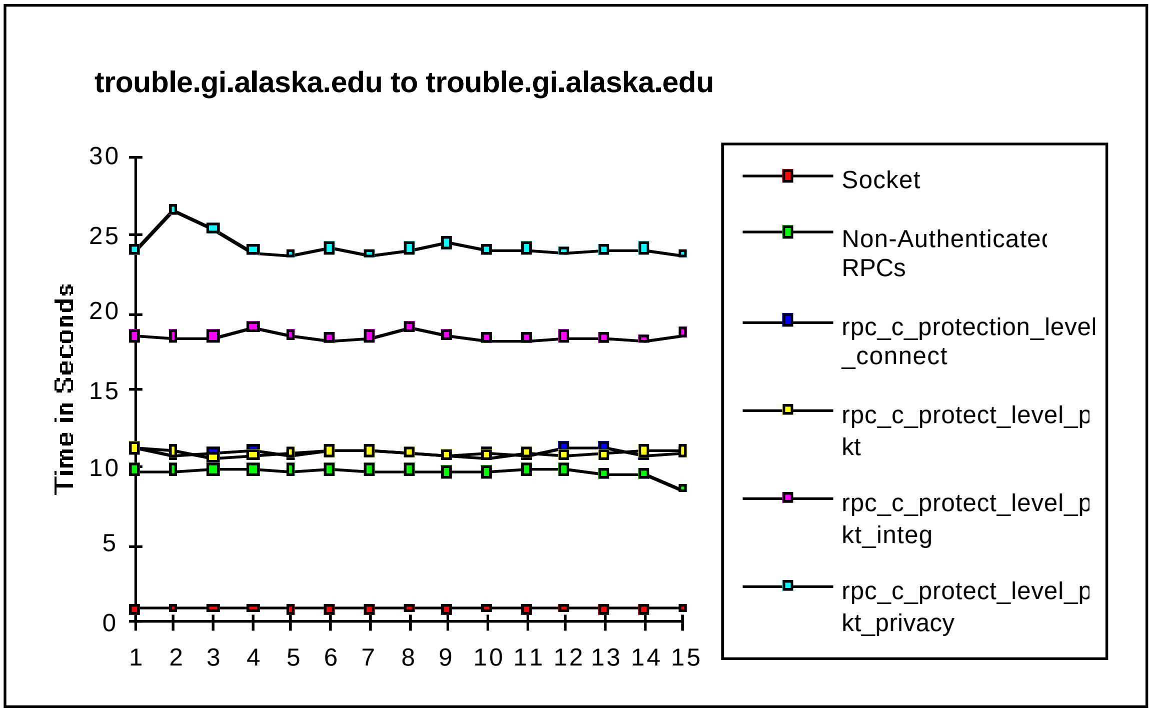


Figure A-19. trouble.gi.alaska.edu to trouble.gi.alaska.edu